Lecture 3

Basic sorting algorithms

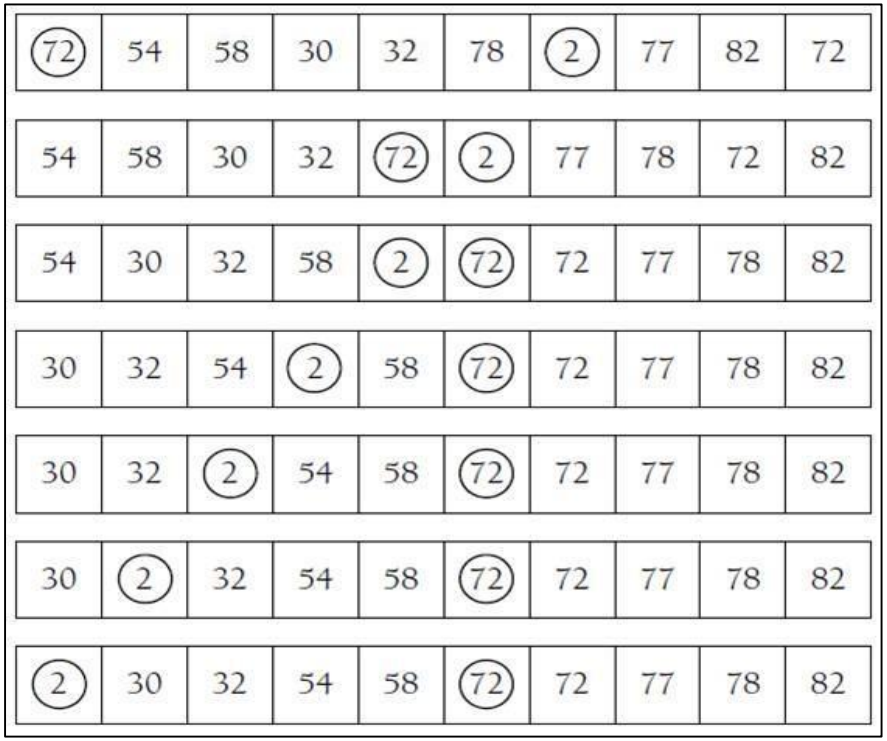
Array filling with random numbers

```
//Example 1
using System;

namespace BasicSorting
{
    class Example1
    {
        static void Main()
        {
            int[] nums = new int[10];
            Random rnd = new Random();
            for (int i = 0; i < 10; i++)
                nums[i]=rnd.Next();
            DisplayIntArray(nums);
            Console.WriteLine("Press any key to continue:");
            Console.Read();
        }
        static void DisplayIntArray(int[] intArray)
        {
            for (int i = 0; i < intArray.Length; i++)
            {
                Console.WriteLine("Element {0} is {1}", i, intArray[i]);
            }
        }
    }
}
```

Bubble sort

- is one of the simplest sorts to understand and implement
- Is one of the slowest sorting algorithms available
- values “float like a bubble” from one end of the list to another



(72)	54	58	30	32	78	(2)	77	82	72
54	58	30	32	(72)	(2)	77	78	72	82
54	30	32	58	(2)	(72)	72	77	78	82
30	32	54	(2)	58	(72)	72	77	78	82
30	32	(2)	54	58	(72)	72	77	78	82
30	(2)	32	54	58	(72)	72	77	78	82
(2)	30	32	54	58	(72)	72	77	78	82

- The code to swap two array elements is written inline rather than as a subroutine
- Simply, from the start of the array, compare each adjacent two numbers. If they are in the desired order leave them as they are. Otherwise, swamp them.
- After each pass, there is one number that took its correct place, so there is no need to pass by it again.
- Why upper ends at 1? Why inner stops at outer-1?
- Is it descending or ascending order? How to change that?

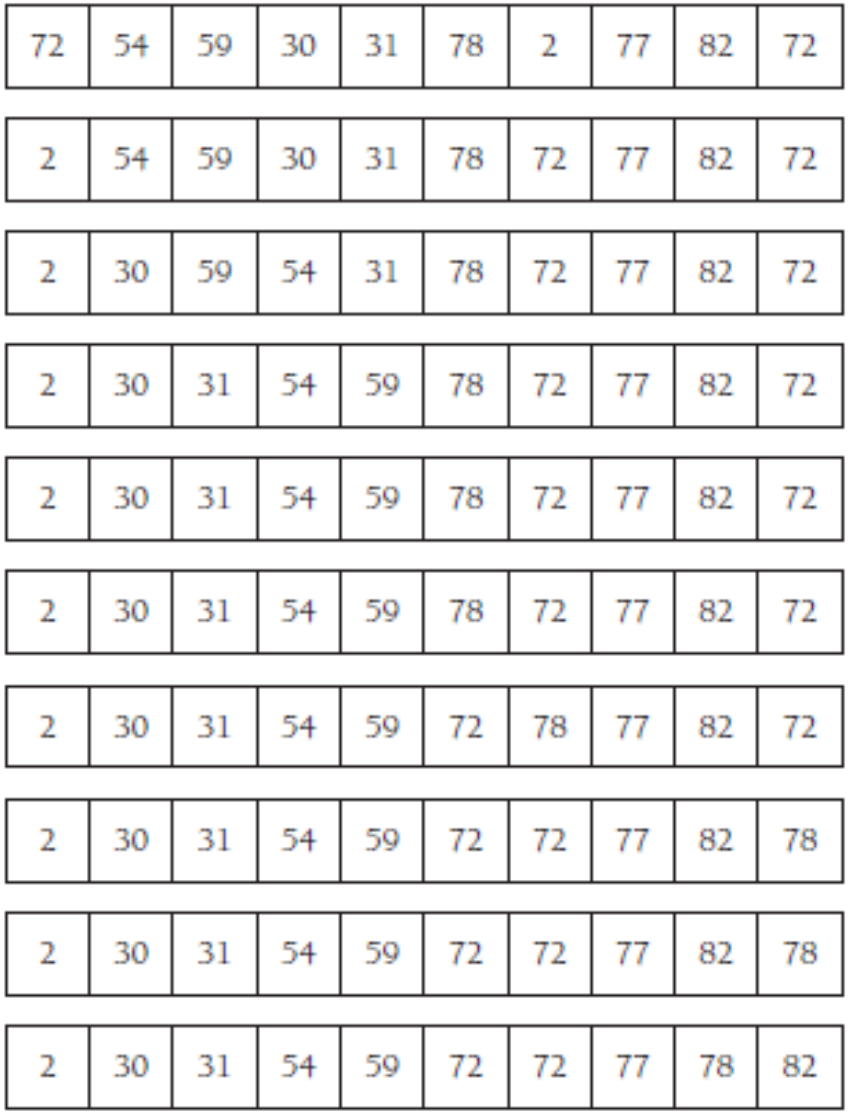
Bubble Sort

- Simply, from the start of the array, compare each adjacent two numbers. If they are in the desired order leave them as they are. Otherwise, swamp them.
- The code to swap two array elements is written inline rather than as a subroutine
- After each pass, there is one number that took its correct place, so there is no need to pass by it again.
- Why outer ends at 1? Why inner stops at outer-1?
- Is it descending or ascending order? How to change that?

```
1   // example 2
2   using System;
3   namespace BasicSorting
4   {
5       class Example2
6       {
7           static void Main()
8           {
9               int[] nums = new int[10];
10              Random rnd = new Random();
11              for (int i = 0; i < 10; i++)
12                  nums[i] = rnd.Next(100);
13              Console.WriteLine("The unsorrted array: ");
14              DisplayIntArray(nums);
15              Console.WriteLine("The sorted array using bubble sort: ");
16              BubbleSort(nums);
17              DisplayIntArray(nums);
18              Console.WriteLine("Press any key to continue:");
19              Console.Read();
20          }
21          static void BubbleSort(int[] intArray)
22          {
23              int temp;
24              for (int outer = intArray.Length-1; outer >= 1; outer--)
25              {
26                  for (int inner = 0; inner < outer; inner++)
27                      if (intArray[inner] > intArray[inner + 1])
28                      {
29                          temp = intArray[inner];
30                          intArray[inner] = intArray[inner + 1];
31                          intArray[inner + 1] = temp;
32                      }
33              }
34          }
35          static void DisplayIntArray(int[] intArray)
36          {
37              for (int i = 0; i < intArray.Length; i++)
38              {
39                  Console.WriteLine("Element {0} is {1}", i, intArray[i]);
40              }
41          }
42      }
```

Selection sort

- starting at the beginning of the array, comparing the first element with the other elements. If a smaller one found, swap it with the first location
- the sort then begins again at position 2
- This continues until each position, except the last position, has been the starting point for a new loop



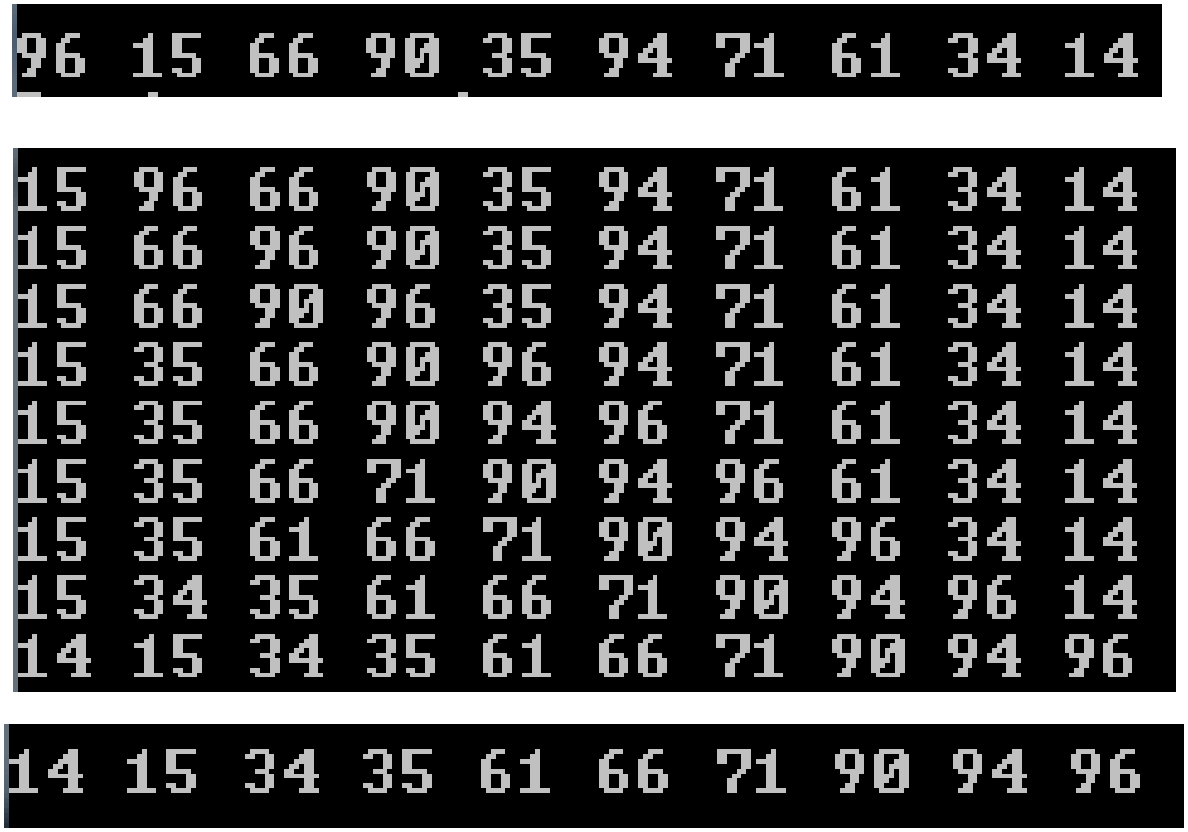
Selection Sort

- The outer loop moves from the first element in the array to the last element
- The inner loop moves from the outer +1 element of the array to the last element, looking for value that are smaller than the element currently being pointed at by the outer loop
- After each iteration of the inner loop, the most minimum value in the reset of the array is assigned to its proper place in the array
- Is it descending or ascending order? How to change that?

```
using System;
namespace BasicSorting
{
    class Example3
    {
        static void Main()
        {
            int[] nums = new int[10];
            Random rnd = new Random();
            for (int i = 0; i < 10; i++)
                nums[i] = rnd.Next(100);
            Console.WriteLine("The unsorrted array: ");
            DisplayIntArray(nums);
            Console.WriteLine("The sorted array using bubble sort: ");
            SelectionSort(nums);
            DisplayIntArray(nums);
            Console.WriteLine("Press any key to continue:");
            Console.Read();
        }
        static void SelectionSort(int[] intArray)
        {
            int min, temp;
            for (int outer = 0; outer < intArray.Length-1; outer++)
            {
                min = outer;
                for (int inner = outer + 1; inner < intArray.Length; inner++)
                    if (intArray[inner] < intArray[min])
                        min = inner;
                temp = intArray[outer];
                intArray[outer] = intArray[min];
                intArray[min] = temp;
            }
        }
        static void DisplayIntArray(int[] intArray)
        {
            for (int i = 0; i < intArray.Length; i++)
            {
                Console.WriteLine("Element {0} is {1}", i, intArray[i]);
            }
        }
    }
}
```

Insertion sort

- starting at the second element in the array:
- Save a copy of the element
- Move the preceding elements one position down until finding the correct position for the saved element
- Put the saved element in the correct position



Insertion Sort

- The outer loop moves from the second element in the array to the last element: The element at the outer loop index is saved
- The inner loop moves from the outer loop element of the array to the second element, looking for the element greater than or equal to the element currently being pointed at by the outer loop
- After each iteration of the inner loop, the most minimum value in the reset of the array is assigned to its proper place in the array
- Is it descending or ascending order? How to change that?

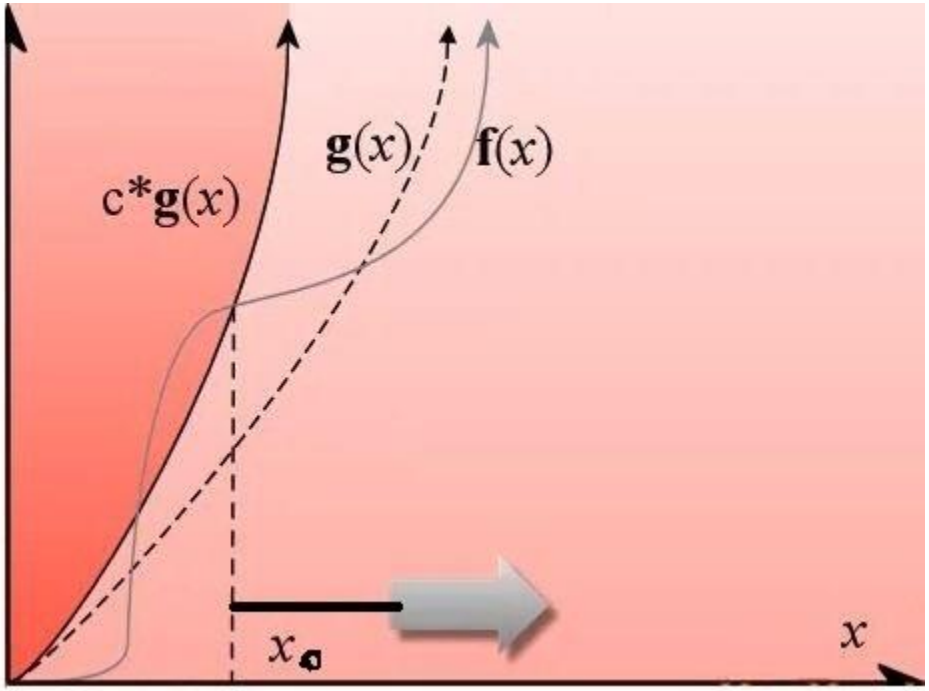
```
using System;
namespace BasicSorting
{
    class Example4
    {
        static void Main()
        {
            int[] nums = new int[10];
            Random rnd = new Random();
            for (int i = 0; i < 10; i++)
                nums[i] = rnd.Next(100);
            Console.WriteLine("The unsorrted array: ");
            DisplayIntArray(nums);
            Console.WriteLine("The sorted array using insertion sort: ");
            InsertionSort(nums);
            DisplayIntArray(nums);
            Console.WriteLine("Press any key to continue:");
            Console.Read();
        }
        static void InsertionSort(int[] intArray)
        {
            int inner, temp;
            for (int outer = 1; outer < intArray.Length; outer++)
            {
                temp = intArray[outer];
                inner = outer;
                while (inner > 0 && intArray[inner - 1] >= temp)
                {
                    intArray[inner] = intArray[inner - 1];
                    inner -= 1;
                }
                intArray[inner] = temp;
            }
        }
        static void DisplayIntArray(int[] intArray)
        {
            for (int i = 0; i < intArray.Length; i++)
            {
                Console.WriteLine("Element {0} is {1}", i, intArray[i]);
            }
        }
    }
```

Report Discussion

Last lecture report: .NET List(of T) methods and properties

New report: Time comparison between the three basic sorting algorithms on different array sizes

Big-O analysis



Definition of big-O

$$f(x)\ is\ O\big(g(x)\big)\ or\ f(x) \in O\big(g(x)\big)$$

If there is exist C and $x_0 > 0$ **such that** $f(x) \leq Cg(x)\ for\ all\ x \geq x_0$

Big-O Example

$$T(n) = \frac{3}{2}n^2 + \frac{5}{2}n - 3$$

By try and error plus intuition (function graphs can be helpful) we can find a constant C=7 and $n_0 = 1$ so that we have $Cg(n) = 7n^2$

$$T(n) = \frac{3}{2}n^2 + \frac{5}{2}n - 3 \; < Cg(n), \qquad for\ all\ n > 1$$

$$T(n) = \frac{3}{2}n^2 + \frac{5}{2}n - 3 \; < 7n^2, \qquad for\ all\ n > 1$$

$$so \quad T(n)\ is\ O(n^2)$$

Note that many other constants can go, but the value of the constant is not important for large n (asymptotic analysis) so any constant that satisfies the condition can go. Also note that we could say Cg(n)= $7n^3$ instead of saying Cg(n)= $7n^3$ but $7n^2$ is the simple function form that satisfies the condition and gives the least difference $T(n)$ and g(n). As the asymptotic function gives small differences from the evaluated function as it expresses more the asymptote.

Big-O rules 1

For all the following theorems, assume that f(n) is a function of n and that K is an arbitrary constant.

Theorem 1: K is O(1)

Theorem 2: A polynomial is O(the term containing the highest power of n)

$$f(n) = 7n^4 + 3n^2 + 5n + 1000 \text{ is } O(7n^4)$$

Theorem 3: K*f(n) is O(f(n)) [i.e., constant coefficients can be dropped]

$$g(n) = 7n^4 \text{ is } O(n^4)$$

Theorem 4: If f(n) is O(g(n)) and g(n) is O(h(n)) then f(n) is O(h(n)). [transitivity]

$$f(n) = 7n^4 + 3n^2 + 5n + 1000 \text{ is } O(n^4)$$

Big-O rules 2

Theorem 5: Each of the following functions is big-O of its successors:

- K [constant]
- $\log_b(n)$ [always log base 2 if no base is shown]
- n
- $n \log_b(n)$
- n^2
- n to higher powers
- 2^n
- 3^n
- larger constants to the n-th power
- n! [n factorial]
- n^n

$f(n) = 3n\log(n)$ is $O(n\log(n))$ and $O(n^2)$ and $O(2^n)$

Big-O rules 3

Theorem 6: In general, f(n) is big-O of the dominant term of f(n), where "dominant" may usually be determined from Theorem 5.

$$f(n) = 7n^2 + 3n\log(n) + 5n + 1000 \text{ is } O(n^2)$$

$$g(n) = 7n^4 + 3^n + 1000000 \text{ is } O(3^n)$$

$$h(n) = 7n(n + \log(n)) \text{ is } O(n^2)$$

Theorem 7: For any base b, $\log_b(n)$ is O(log(n)).

Big-O of the three basic sorting algorithms

Bubble sort

The worst statement is the comparison statement. It executes $f(n) = (n-1)(1+2+\cdots+(n-1) = (n-1)+2(n-1)+\cdots+(n-1)(n-1) = k+\alpha n+\gamma n^2$

$F(n)\ is\ O(n^2)$ Bubble sort is $O(n^2)$

Selection sort

The worst statement is the comparison statement. It executes $f(n) = (n)((n-1)+(n-2)+\cdots+\cdots+1 = k+\alpha n+\gamma n^2$

$F(n)\ is\ O(n^2)$ Selection sort is $O(n^2)$

Insertion sort

The worst statement is the comparison statement. In the worst cases, it executes $f(n) = (n-1)(1+2+\cdots+(n-1) = (n-1)+2(n-1)+\cdots+(n-1)(n-1) = k+\alpha n+\gamma n^2$

$F(n)\ is\ O(n^2)$ Insertion sort is $O(n^2)$

Also the three algorithms gives the same big-O asymptotically, the insertion sort is the worst one because the worst operation is likely to execute much in this algorithm. However, asymptotically, they have all the same complexity.